Pointers

A pointer is a variable that contains the address of another variable. It is a derived data type.
Pointers are the most powerful features of language.
A pointer can also be used to refer to another pointer function. The purpose of pointer is to save memory space and achieve faster execution time.

- If variable is of integer type the pointer must be integer.
- If variable is float type the pointer must be float
- If variable is char type the pointer must be char.

A pointer can contain the address of any basic data type, of array, structures, unions etc.
Pointers contain addresses not value.

Advantages:

1. Pointers allow, return more than one value from a function.
2. We can allocate or reallocate space in memory by using pointers.
3. It allows passing of arrays and strings to functions.
4. Pointers are used to implement several data structures like Stacks, Queues, and Linked lists.
5. With the use of pointers program execution becomes faster.
6. Using pointers, arrays and structures can be handled in more efficient way.

How to get the address of a variable?

We can get the address of any variable using a unary operator called **Address of** operator **&**. As the address of any variable is a positive number, the format specifies %u or %p or %x is used while printing the address. %u prints the address in decimal, %p and %x print the addresses in hexadecimal.

Address operator: & is known as address operator, The address operator returns the address of a variable.It is an unary operator.
Example: WAP to show the use of "address of operator “

```
#include<stdio.h>
#include <conio.h>
void main()
{
int var=100;
clrscr();
printf(“the address of var is %u”,&var);
getch();
}
```

Declaring pointer variables:

Like other variables in C, a pointer variable has to be declared before it can be used in the program.

Syntax: data type * name;
Where "data type" is the type of the pointer variable and "name" is the name of the pointer variable. The asterisk Sign (*) is used to inform the compiler that the variable being is a pointer.
Example: int *ptr;

Initialization of pointer:

Pointer will contain garbage value if we don't initialize them. So it is necessary to initialize the pointer variables properly. The pointer variables are initialized using address of operator.
Example: int a=90;
int *ptr &a;

/*WAP using pointer*/

```
#include<stdıo.h>
#include<conio.h>
void main()
{
int x=10;
int *ptr;
ptr &x;
clrscr():
printf("\n vale of x=%d",x);
printf("\n value of ptr=%u",ptr);
printf("\n address of ptr=%u",&ptr);
printf("\n value of ptr=%d". *ptr);
getch():
}
```

ACCESSING VARIABLE THROUGH ITS POINTER

After initialization pointer variable, if we want to access the value of pointer, we have to put an **asterisk** ('*' sign) character just before the pointer variable.

indirection Operator: The '*' is an indirection operator. It is a unary operator. It is used to create a pointer variable.
Indirection operator is used to find out value.

```
#include<stdıo.h>
#include<conio.h>
void main()
{
clrscr();
int num = 100;
int *ptr;
ptr = &num;
printf("value of num = %d\n", *ptr);
getch();
}
```

Pointer and Arrays

Pointer and array are very closely related to each other. An array is group of elements of same type. In C the name of an array identify as base address of the array.

Pointer and 1-D Array

The array will be stored in the contiguous memory location. Let us consider the following array declaration:-

```
int ar[]={10,20,30,40,50};
```

10	20	30	40	50
2001	2003	2005	2007	2009

The array ar is int type, each element of this array occupies two bytes. The name of the array contains the starting address of the array.

Example: WAP to show the relation between array and pointer

```
#include<stdio.h>
#include<conio.h>
void main()
{
int ar[5]={10,20,30,40,50);
int *p,i;
p=ar;
for (i=0;i< 5; i++)
{
printf("\n the value stored at the location %d is %d", i,p[i]);
```

```
}
getch():
}
```

Output:

```
the value stored at the location 0 is 10
the value stored at the location 1 is 20
the value stored at the location 2 is 30
the value stored at the location 3 is 40
the value stored at the location 4 is 50
```

Pointer and 2 D Array

In 2 dimensional array , another index variable j is used. Let us consider the following array declaration:-

```
int ar[3][4];
```

- Where ar is name of two-dimensional array of 3 rows and 4 columns.
- E.g. int ar[3][4] can be stored in the contiguous memory location.

To access the elements of a 2-dim array

```
a[i][j]=*(*(a+i)+j)
```

/* WAP 2 D Array with pointer*/

```
#include<stdio.h>
#include<conio.h>
void main()
{
int ar[3][4]={
{10,20,30,40}, {5,6,7,8}, {9,10,11,12},
};
int i,j;
clrscr();
for (i=0;i<3;i++)
{
for(j=0;j<4;j++)
{
printf("\n The value stored at the location[%d][%d]is %d", i, j, *(*(ar+i) +j));
}
getch();
}
```